UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/074,705	02/13/2002	Eric M. Dowling	SEARCHP.011C1DV2	8133

27299 7590 07/25/2005

GAZDZINSKI & ASSOCIATES
11440 WEST BERNARDO COURT, SUITE 375
SAN DIEGO, CA 92127

EXAMINER	
HUISMAN, DAVID J	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 07/25/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/074,705	DOWLING, ERIC M.	
	Examiner	Art Unit	
	David J. Huisman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *16 May 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-43* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) *2 and 42* is/are allowed.

6)☒ Claim(s) *1,3-41 and 43* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *16 May 2005* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-43 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Amendment as received on 5/16/2005.

Supervisor Interview Request

3. The examiner has noted that applicant requested an interview with the examiner's supervisor for the reasons set forth in the remarks. However, it is not clear what applicant wishes to discuss. If applicant wishes to discuss the prior art of record and how it does or does not read on applicant's claims, then the examiner feels that this could advance prosecution, and an interview could be granted. However, if applicant merely wishes to argue the use of Official Notice by the examiner and/or the fact that the claims were previously indicated as allowable by the previous examiner, the examiner feels that an interview would not advance prosecution, and therefore, would be unnecessary (MPEP 713). If applicant still wishes to set up an interview, it is asked that applicant either file an "applicant initiated interview request" form (PTOL-413) including a brief description of what is to be discussed (MPEP 713) or applicant may simply call the examiner at the phone number listed on the last page of this Office Action so that a submission of an agenda (via fax) and an interview time could be set up over the phone.

Specification

4. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed. From MPEP 606.01[R-2], "This may result in slightly longer titles, but the loss in brevity of title will be more than offset by the gain in its informative value in indexing, classifying, searching, etc. If a satisfactory title is not supplied by the applicant, the examiner may, at the time of allowance, change the title by examiner's amendment."

5. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

Claim Objections

6. Claim 3 is objected to because of the following informalities: On page 6, lines 12-13, the phrase "prefetch time-critical the data to be used" is grammatically incorrect. On page 6, line 13, the phrase "operands one or more" is grammatically incorrect. Finally, on page 6, line 15, the phrase "the required the data" is grammatically incorrect. Appropriate correction is required.

7. Claim 40 is objected to because of the following informalities: On page 16, line 18, replace "load" with --loadable--. Appropriate correction is required.

8. Due to the length of the application, the examiner asks applicant's cooperation in finding all minor errors within the claims. The examiner has done his best in finding most of them, but applicant should review the claims and make sure no other errors exist.

Claim Rejections - 35 USC § 112

9. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

10. Claim 40 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

11. Claim 40 recites the limitations "said parallel-loadable register file" on page 16, lines 19-20. There is insufficient antecedent basis for these limitations in the claim because it is not clear whether applicant is referring to the first or second parallel-loadable register files (page 16, line 10).

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. Claims 1, 24, 27, and 35-36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Parady, U.S. Patent No. 5,933,627 (as disclosed by applicant), in view of Inagami et al., U.S. Patent No. 4,881,168 (as disclosed by applicant and herein referred to as Inagami).

14. Referring to claim 1, Parady has taught a method comprising the steps of:

a) segmenting the architecture into first and second portions. It should be noted that any group of components within Parady may perform a portion. Therefore, first and second portions would exist.

b) executing instructions by said first portion which manipulate only register operands. See Fig.1 and note that components 38 and 40, for instance, may be considered part of a first portion. These two components would execute floating-point addition, subtraction, and multiplication instructions, which manipulate register only operands.

c) Parady has taught executing instructions by said second portion which perform row-oriented load/store operations (see Fig.1, component 32, and note that the load/store unit inherently performs load/store instructions from/to rows of memory). Parady has not explicitly taught individual register-to-register move operations. However, Official Notice is taken the register-register move operations are well known and expected in the art. These operations at the very least allow for the copying of registers. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement register-register move operations.

d) at a given time, said first portion of said architecture sees a subset of the total available registers as its set of architectural registers while a second subset of said total available registers is not accessible by said first portion of said first architecture. See Fig.3, components 48 and 50, and note that only one thread is in execution at any given time. Consequently, only that thread's corresponding registers are available to the portion. All other registers form a second set which are inaccessible because they belong to threads which are not executing. These corresponding registers are merely a subset of all available registers.

e) said first portion of said architecture comprises one or more functional units which execute a first program comprising instructions using register operands. Again, see Fig.1 and note that components 38 and 40, for instance, may be considered part of a first portion.

f) Parady has not taught that said second portion of said architecture executes a second program tightly coupled to said first program, said second program comprising parallel row-oriented load/store/mask commands. However, Inagami has taught performing such operations. See Fig.2, 3A, 3B, 4, and 5. And, as disclosed in column 1, lines 42-48, having such instructions allow the loading/storing of certain data elements, thereby allowing for more efficient processing of some programs. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady to include these row-oriented load/store/mask commands. Furthermore, Parady has explicitly taught register-to-register move commands but as described above, this would have been an obvious modification. Finally, Parady has taught architectural register set switch commands to insure that data accessed by said first program is available when it is needed. See the abstract. Note that when a particular instruction misses a cache, for instance, a thread switch will occur, thereby causing the next thread's register file to become active (which insures that data accessed by that thread is available when needed).

15. Referring to claim 24, Parady has taught a processor as described in claim 21. Parady has not taught that the data assembly further comprises a bit mask to select one or more data locations within at least one of said register sets and an instruction set which comprises a command to load a set of selected elements of a row said DRAM array into a selected set of data registers, said selection based on at least one bit in said bit mask. However, Inagami has taught performing such operations. See Fig.2, 3A, 3B, 4, and 5. And, as disclosed in column 1, lines 42-48, having such instructions allow the loading/storing of certain data elements, thereby allowing for more efficient processing of some programs. Consequently, it would have been

obvious to one of ordinary skill in the art at the time of the invention to modify Parady to include these row-oriented load/store/mask commands.

16. Referring to claim 27, Parady has taught a processor as described in claim 21. Parady has not taught a mask and switch unit interposed between said DRAM array and at least one of said functional units. However, Inagami has taught such a unit which performs load/store/mask operations. See Fig.2, 3A, 3B, 4, and 5. As disclosed in column 1, lines 42-48, having such a unit and instructions allows the loading/storing of certain data elements, thereby allowing for more efficient processing of some programs. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady to include a mask/switch unit for performing these row-oriented load/store/mask commands.

17. Referring to claim 35, Parady has taught a method as described in claim 33. Parady has not taught that said speculative loading is further controlled by a bit mask that identifies a selected subset of said register file to be speculatively loaded. However, Inagami has taught loading based on a mask. See Fig.2, 3A, 3B, 4, and 5. As disclosed in column 1, lines 42-48, having such a unit and instructions allows the loading/storing of certain data elements, thereby allowing for more efficient processing of some programs. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady to include a mask unit for performing these row-oriented load/mask commands.

18. Referring to claim 36, Parady has taught a method as described in claim 33. Parady has not taught that said speculative loading is further processed via a mask and switch unit whereby a selected subset of said register file is speculatively loaded with a data element order permutation. However, Inagami has taught such a unit which performs load/store/mask operations. See Fig.2,

3A, 3B, 4, and 5. As disclosed in column 1, lines 42-48, having such a unit and instructions allows the loading/storing of certain data elements, thereby allowing for more efficient processing of some programs. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady to include a mask/switch unit for performing these row-oriented load/store/mask commands.

19. Claims 3 and 43 are rejected under 35 U.S.C. 103(a) as being unpatentable over Inagami, as applied above, in view of Hennessy and Patterson, "Computer Architecture - A Quantitative Approach, 2nd Edition" 1996 (herein referred to as Hennessy), in view of Hayashi et al., U.S. Patent No. 5,237,702 (herein referred to as Hayashi), and further in view of Jager, U.S. Patent No. 5,423,048.

20. Referring to claim 3, Inagami has taught in a processor comprising a plurality of memory arrays comprising rows and columns of random access memory cells (Fig.1, component 1), a set of functional units which execute a first program (Fig.1, component 5), and a data assembly unit which executes a second program (Fig.1, components 2-0, 2-1, etc.), said second program being tightly coupled with said first program, and whereby said data assembly unit is operative to load and store a plurality of data elements from a memory row of a main memory comprising at least one array to or from one or more register files (Fig.1, components VR0, VR1,....VR7) which each include a parallel access port, a method of intelligent caching comprising the steps of:

a) Inagami has not explicitly taught dispatching in parallel a first sequence of instructions and a second sequence of instructions. However, see Hennessy, pp.278-280, and note that when a system has separate functional units, each one can execute a different instruction per clock cycle

(this is a superscalar system). For instance, see figure 4.26 on page 280 and note the existence of an integer unit and a floating-point unit, which are independent of each other. Consequently, they may execute instructions independently in the same cycle, thereby increasing throughput (executing two instructions per cycle instead of one instruction per cycle yields higher throughput). Similarly, Inagami teaches separate load/store pipes 2 and execution pipes 5, just like the integer unit and floating-point unit of Hennessy is separate. The independence of units allows for concurrent execution. Therefore, because Inagami is suited for concurrent execution, in order to increase throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to concurrently dispatch and execute first and second sequences of instructions.

b) executing the first sequence of instructions on said set of functional units, said functional units operative to process data stored in said register files. See column 4, line 63, to column 5, line 1. note that arithmetic instruction would be executed here.

c) executing a second sequence of instructions on said data assembly unit, said data assembly unit operative to transfer data between said register files and main memory. See Fig.1 and note that data is transferred between registers and memory via the load/store pipes (data assembly unit).

d) Inagami has not taught that while said first sequence of instructions is being processed, in advance to a time when one or more particular instructions in the first sequence will need to use corresponding particular data as operands, said second sequence of instructions instructs said data assembly unit to prefetch the particular data to be used as data operands to the one or more particular instructions in the first sequence of instructions into said register files from said at

least one DRAM array via said parallel access port. However, Hayashi has taught such a concept. See the title and claim 1 of Hayashi and note that vector data is prefetched and ultimately written to vector registers. Clearly, prefetching is beneficial because by bringing data into the system from memory before instructions requiring that data are executed, once the instruction does in fact need to execute, the data is available immediately, thereby preventing a time delay. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to include prefetching capability.

e) Inagami in view of Hayashi has not taught that when conditional logic in said first program makes it uncertain as to which data will next be needed by said functional units executing said first sequence of instructions, said second sequence of instructions instructs said data assembly unit to prefetch time-critical data to be used as data operands one or more particular instructions in the first sequence of instructions so that irrespective of the conditional outcome in processing said first sequence of instructions, the required data to be used as data operands to the one or more particular instructions in the first sequence of instructions will be present in said registers. However, Jager has taught that it is beneficial to prefetch along both branch paths because the required data will be available regardless of the final direction of the branch. This is contrary to prefetching along a single path, where if the path turns out to be the wrong path, the prefetching was not helpful and efficiency is reduced. See the abstract and column 1, line 50, to column 2, line 27. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami in view of Hayashi to include prefetching along both paths of a branch (which are well known instructions) so that data will always be available.

f) Inagami has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.

21. Referring to claim 43, Inagami in view of Hennessy in view of Hayashi and further in view of Jager has taught a method as described in claim 3. The combination has further taught that waiting for the particular data to be loaded is avoided by having the particular data available in the register file and accessible by the first program at the time the particular instructions are executed. This is the inherent nature of prefetching. That is the whole idea of prefetching is to fetch data ahead of when it is needed to that it is available immediately when it is needed. This avoids delays because the system will not have to wait for the data to be fetched right when it is needed.

22. Claims 4, 12, 20, 29-32, and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Parady, as applied above, in view of Bissett et al., U.S. Patent No. 5,896,523 (herein referred to as Bissett).

23. Referring to claim 4, Parady has taught in a processor comprising a plurality of memory arrays (Fig.2, main memory) which comprise rows and columns of random access memory cells, a set of functional units that execute a first program (see Fig.1, components 38, 40, 42, for instance), and a data assembly unit that executes a second program (see Fig.1, component 32, for

instance), said second program being tightly coupled with said first program, and wherein said data assembly unit is operative to load and store a plurality of data elements from a DRAM row to or from one or more register files which each include a parallel access port (see Fig.1, components 48 and 40 and note the multiple access ports for parallel access), and a selector switch operative to include or remove a register file from the architectural register set of said functional units executing said first sequence of instructions (see the abstract, column 2, lines 35-37, and Fig.3, component 112, and note that when a thread is switched out, it's corresponding register file becomes inactive (removed), a method of intelligent caching comprising the steps of:

a) dispatching in parallel said first sequence of instructions and a second sequence of instructions. See Fig.1 and column 3, lines 11-17, and note that multiple instructions are dispatched concurrently to multiple functional units (which include the load/store unit, floating-point units, etc.)

b) executing said first sequence of instructions on said functional units, whereby said instructions involve operands, and said operands correspond to architectural registers visible to said functional units. Clearly, the floating-point functional units will execute floating-point instructions using the floating-point registers as operands.

c) executing said second sequence of instructions on said data assembly unit. Loads and stores are executed on the data assembly unit (load/store unit).

d) Parady has not taught that said execution of said second sequence of instructions is operative to prefetch information into one or more register files which are not architectural registers visible to said functional units. However, Bissett has taught performing data prefetches in the background. And, this is beneficial because background operations do not influence software

execution, thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady in view of Bissett such that Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread).

e) in response to progress made in said first program, said data assembly unit executing one or more instructions which transform said one or more register files which received prefetched data into architectural register files visible to said functional units and transform current architectural register files into non-architectural register files which are inaccessible to said functional units. As described above, when an instruction in a first thread missed the cache (see the abstract), a thread is switched, thereby also causing a new register file to become active.

f) Parady has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady's main storage to be a DRAM array.

24. Referring to claim 12, Parady has taught in a processor comprising a plurality of memory arrays (Fig.2, main memory) which comprise rows and columns of random access memory cells,

at least one functional unit that executes a first sequence of instructions (see Fig. 1, components 38, 40, 42, for instance), and a data assembly unit that executes a second sequence of instructions (see Fig. 1, component 32, for instance), said second sequence of instructions being tightly coupled with said first sequence of instructions, and whereby said data assembly unit is operative to cause a plurality of data elements to be transferred in parallel between a memory row and one or more register files via a parallel access port in each of said register files (see Fig. 1, components 48 and 40 and note the multiple access ports for parallel access), a method of intelligent caching comprising the steps of:

a) executing said first sequence of instructions on said at least one functional unit, whereby said first sequence of instructions involve operands, and said operands correspond to architectural registers visible to said at least one functional unit. Clearly, the floating-point functional units will execute floating-point instructions using the floating-point registers as operands.

b) executing said second sequence of instructions on said data assembly unit. Loads and stores are executed on the data assembly unit (load/store unit).

c) Parady has not taught that said execution of said second sequence of instructions is operative to prefetch information into one or more register files which are not architectural registers visible to said functional units. However, Bissett has taught performing data prefetches in the background. And, this is beneficial because background operations do not influence software execution, thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to

execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady in view of Bissett such that Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread). It can be seen that the execution of the second sequence would cause this prefetching because while the second sequence is executing, prefetching should occur for an inactive thread. If the second sequence belongs to thread 1, then prefetching for a thread other than thread 1 should occur.

d) executing one or more instructions which transform at least one of said one or more register files into an architectural register file visible to said at least one functional unit. As described above, when an instruction in a first thread missed the cache (see the abstract), a thread is switched, thereby also causing a new register file to become active.

e) Parady has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady's main storage to be a DRAM array.

25. Referring to claim 20, Parady has taught a processor as described in claim 19. Parady has further taught:

a) each said register file is capable of being placed into an active state and an inactive state. From column 2, lines 25-39 and Fig.3 of Parady, it should be realized that each thread has its

own register file. Consequently, when a thread is running, its corresponding register file is the only active register file; the rest are inactive.

b) said functional unit is responsive to commands involving architectural register operands that map onto to the registers within a register file that is in the active state. Clearly, the system of Fig.1 would operate in such a manner. For instance, component 34 of Fig.1 could execute a multiplication of two operands (retrieved from the register file). And, the current instructions will be of a current thread which has a corresponding active register file.

c) Parady has not taught that said selected one of said register files is in the inactive state (i.e., Parady has not taught that prefetching has occurred for an inactive register file). However, Bissett has taught performing data prefetches in the background. And, this is beneficial because background operations do not influence software execution, thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady in view of Bissett such that Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread).

26. Referring to claim 29, Parady has taught an intelligent-cache based processor comprising:

a) a memory array comprising a plurality of random access memory cells. See Fig.2 and note that the system is coupled to main memory which inherently has rows and columns of memory cells. Parady has not explicitly taught that the memory array is a DRAM array. However,

Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady's main storage to be a DRAM array.

b) first and second dual-port registers files, whereby the first port of each of said register files is a parallel access port and is coupled in parallel to said DRAM array, each said of register files is capable of being placed into an active state and an inactive state. See Fig.1, and note that each register file 48 and 50 is a dual port file (has port for reading and port for writing). The first port would be the write port which is coupled to memory for loading data into the file. It would also be a parallel access port because the write port is in fact 3 write ports, according to Fig.1, so 3 reads may occur in parallel. The second port would be the read port so that functional units may read data and operate on it. Finally, from column 2, lines 25-39 and Fig.3 of Parady, it should be realized that each thread has its own register file. Consequently, when a thread is running, its corresponding register file is the only active register file; the rest are inactive.

c) at least one functional unit that executes a first program, said functional unit coupled to said second port of said register files, said functional unit responsive to commands involving architectural register operands that map onto to the registers within a register file that is in the active state. Clearly, the system of Fig.1 would operate in such a manner. For instance, component 34 of Fig.1 could execute a multiplication of two operands (retrieved from the register file) and then write the result to the register file via the second port. And, the current instructions will be of a current thread which has a corresponding active register file.

d) Parady has not explicitly taught a data assembly unit that executes an intelligent caching program in support of said first program, said data assembly unit responsive to at least one command that causes data to be moved between the DRAM array and a register file that is in the inactive state. However, Bissett has taught performing data prefetches in the background. And, this is beneficial because background operations do not influence software execution, thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady in view of Bissett such that Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread).

e) whereby the first and second register files are capable of toggling between said active and inactive states, under program control, during program execution. As described in column 2, lines 25-39, each thread has its own register file (Fig.3). And, only one thread is active at a time, where threads switch due to a long latency event. See the abstract. Consequently, if thread 1 is executing, then thread 1's register file will be in the active state while the other thread's register files will be in the inactive state. However, when thread 1 becomes inactive, so will its register file and another thread and its register file will become active.

27. Referring to claim 30, Parady in view of Bissett has taught a processor as described in claim 29. Parady has further taught a control coupling between said data assembly unit and said register files, whereby said data assembly unit executes an instruction set which comprises a

command to cause at least one of the register files to toggle between said active and said inactive state. See the abstract and Fig.3 of Parady and note that in response to a load instruction missing the cache, a thread switch will occur.

28. Referring to claim 31, Parady in view of Bissett has taught a processor as described in claim 29. Parady has further taught a control coupling between said functional unit and said register files, whereby said functional unit executes an instruction set which comprises a command to cause at least one of the register files to toggle between said active and said inactive state.

29. Referring to claim 32, Parady in view of Bissett has taught a processor as described in claim 29. Parady has further taught:

a) said functional unit is responsive to an instruction set, and instructions within said instruction set comprise commands exclusively responsive to register operands, said register operands corresponding to a set of architectural registers. See Fig.1, and note for instance that component 34 will execute multiply and divide instructions. These instructions typically include only register operands and are in the form of MULT R3, R2, R1 (multiply R1 and R2 and store the result in R3).

b) said data assembly unit is responsive to an instruction set, and instructions within said instruction set comprise:

(i) Parady has not taught a command to parallel load at least a portion of an inactive register set from said DRAM array. However, Bissett has taught performing data prefetch commands in the background. And, this is beneficial because background operations do not influence software execution, thereby allowing the current program to

run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady in view of Bissett such that Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread).

(ii) a command to toggle said inactive register set into said active state and at the same time to toggle an active register set into said inactive state, whereby said architectural register set of said functional unit is dependent on the execution of said toggle command. From the abstract, it should be noted that a load instruction would perform such a task. That is, when a load causes a cache miss, the current active thread and register file become inactive while another thread and its register file become active. Once the inactive register file becomes active, it becomes the architectural file which is used in execution until the next switch.

30. Referring to claim 40, Parady has taught in an intelligent-cache based processor:

a) the processor comprising:

(i) at least one memory array comprising rows and columns of random access memory cells. See Fig.2 and note that the system is coupled to main memory which inherently has rows and columns of memory cells. Parady has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its

advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady's main storage to be a DRAM array.

(ii) at least first and second parallel-loadable register files. See Fig.1 and Fig.3, components 48 and 50. Note that by having multiple write ports, they may be loaded in parallel.

(iii) at least one functional unit that executes a first program and interacts with a set of architectural register locations. See Fig.1, and note that component 34 is a functional unit which would interact with registers (to perform multiplication and division on them).

(iv) a data assembly unit that executes an intelligent caching program in support of said first program. See Fig.1, component 32 and 14 and note that these components may be part of a data assembly unit. Loads and stores are executed by this unit to bring data in from cache/main memory and to send data to memory.

b) a method of intelligent caching processing comprising:

(i) in said at least one functional unit, executing a first group instructions, at least some of which have operands that correspond to said architectural register locations, said architectural register locations being mapped to said first parallel-loadable register file. See Fig.1, and note that component 34 is a functional unit which would interact with registers (to perform multiplication and division on them). The register file used will be

the register file associated with the active thread. See column 2, lines 25-39, and the abstract.

(ii) in said data assembly unit, monitoring at least a subset of bits generated by the execution of said first group of instructions, and in response thereto, causing said second parallel-load register file to be parallel loaded from a row of said DRAM array. Note that when a load instruction is decoded and executed, an address and read signals would be monitored by the DRAM (also part of the assembly unit) and used to read data from the DRAM and load it into the appropriate register. Parady has not taught loading a register file while it is in an inactive state. However, Bissett has taught performing data prefetch commands in the background. And, this is beneficial because background operations do not influence software execution, thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady in view of Bissett such that Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread).

(iii) in said data assembly unit, causing said second parallel-loadable register file to be mapped from said inactive state to said set of architectural register locations accessible by said at least one functional unit. From the abstract, note that when a load causes a cache

miss, a thread will be switched, causing a new register file (and thread) to become active. This register file will now be visible to the functional units.

(iv) in said at least one functional unit, executing a second group of instructions at least some of which have operands that correspond said architectural register locations. Again, the units in Fig.1 (components 34-46, for instance) may execute a second group of instructions (from a second thread) which have operands in the register file.

(v) whereby said second group of instruction cause data in said second parallel-loadable register file to be accessed. See Fig.1, and note that if one instruction in the second group is a divide instruction, for instance, then the register file may be accessed to retrieve a dividend and a divisor so that the division may be performed.

31. Claims 5, 7-10, and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Parady in view of Bissett, as applied above, and further in view of Jager, as applied above.

32. Referring to claim 5, Parady in view of Bissett has taught a method as described in claim 4. Parady in view of Bissett has not taught the step of speculatively prefetching information needed by two or more execution paths when a conditional branch in said first instruction sequence makes it ambiguous as to which data will next be needed by said functional units. However, Jager has taught that it is beneficial to prefetch along both branch paths because the required data will be available regardless of the final direction of the branch. This is contrary to prefetching along a single path, where if the path turns out to be the wrong path, the prefetching was not helpful and efficiency is reduced. See the abstract and column 1, line 50, to column 2, line 27. Consequently, it would have been obvious to one of ordinary skill in the art at the time

of the invention to modify Parady in view of Bissett to include prefetching along both paths of a branch (which are well known instructions) so that data will always be available.

33. Referring to claim 7, Parady has taught a processor that is segmented into first and second portions, said first portion comprising a set of functional units and a set of architectural registers accessed thereby (see Fig.1, components 34, 36, and 48, and note that these components may be considered a first portion), said second portion comprising at least one other functional unit and a first inactive register set (see Fig.1, component 32, Fig.3, component 48, and note that each thread has a corresponding register file and when the thread is inactive, its register file is inactive), said other functional unit capable of moving data between a row of an array and said other register set (the other functional unit 32 is a load/store unit which loads registers with data from a memory array), a method of intelligent caching comprising:

a) splitting a single program into first and second parallely-dispatched and executed portions, said first portion of said program executed on said first portion of the architecture, said second portion of said program executed on said second portion of said architecture. See Fig.1 and column 3, lines 11-17, and note that multiple instructions are dispatched concurrently to multiple functional units (which include the load/store unit, floating-point units, etc.). Note also that integer ALU instructions would be executed by the first portion and load/store instructions would be executed by the second portion.

b) Parady has not taught that said second portion of said architecture is operative to prefetch data into said first inactive register set in anticipation of data requirement a by said first portion of said architecture. However, Bissett has taught performing data prefetches in the background. And, this is beneficial because background operations do not influence software execution,

thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady in view of Bissett such that Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread).

c) Parady in view of Bissett has not taught that prior to when said first portion of said architecture executes a conditional branch instruction, said second portion of said architecture prefetches first and second data sets from memory, said first data set being needed when said condition evaluates to true, said second data set being needed when said condition evaluates to false. However, Jager has taught that it is beneficial to prefetch along both branch paths because the required data will be available regardless of the final direction of the branch. This is contrary to prefetching along a single path, where if the path turns out to be the wrong path, the prefetching was not helpful and efficiency is reduced. See the abstract and column 1, line 50, to column 2, line 27. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady in view of Bissett to include prefetching along both paths of a branch (which are well known instructions) so that data will always be available.

d) Finally, Parady has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been

obvious to one of ordinary skill in the art at the time of the invention to modify Parady's main storage to be a DRAM array.

34. Referring to claim 8, Parady in view of Bissett and further in view of Jager has taught a method as described in claim 7. Furthermore:

a) it is inherent that the prefetching operation comprises executing first an instruction that causes a first row of a DRAM array to be loaded into said first inactive register set. Clearly, prefetches do not occur by themselves. Instead, the system must be instructed to perform a prefetch.

b) executing a second instruction that causes a second row of said DRAM array to be loaded into a second inactive register set. This is also inherent, as prefetches do not occur by themselves. Instead, the system must be instructed to perform a prefetch.

c) checking a condition and in response to the checking, executing a command that causes a selected one of said first and second inactive register sets to be activated to an become an architectural register visible to said first portion of said program. See the abstract and Fig.3, component 114, of Parady, and note that when a cache miss occurs (condition), a thread switch occurs, thereby making the current register set inactive and making an inactive register set active (along with its thread).

35. Referring to claim 9, Parady in view of Bissett and further in view of Jager has taught a method as described in claim 7. Furthermore:

a) it is inherent that the prefetching operation comprises executing first an instruction that causes a first row of a DRAM array to be loaded into said first inactive register set. Clearly, prefetches do not occur by themselves. Instead, the system must be instructed to perform a prefetch.

b) executing a second instruction that causes a second row of said DRAM array to be loaded into a second inactive register set. This is also inherent, as prefetches do not occur by themselves. Instead, the system must be instructed to perform a prefetch.

c) checking a condition and in response to the checking, executing a command that causes said architectural register set to assume an inactive state and a selected one of said first and second inactive register sets to be activated to become architectural registers visible to said first portion. See the abstract and Fig.3, component 114, of Parady, and note that when a cache miss occurs (condition), a thread switch occurs, thereby making the current register set inactive and making an inactive register set active (along with its thread).

36. Referring to claim 10, Parady in view of Bissett and further in view of Jager has taught a method as described in claim 7. Parady in view of Bissett and further in view of Jager has not explicitly taught individual register-to-register move operations. However, Official Notice is taken the register-register move operations are well known and expected in the art. These operations at the very least allow for the copying of registers. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement register-register move operations. Also, recall that Bissett has taught prefetching in the background. This is useful so that data will be available immediately to an instruction upon execution. The register-move operation is similar to prefetching in that data is moved to a register ahead of time so that it will be ready for an instruction.

37. Referring to claim 13, Parady in view of Bissett has taught a method as described in claim 12. Furthermore, claim 13 is rejected for the same reasons set forth in the rejection of claim 5.

38. Claims 6, 14-19, 21-23, 25-26, 28, 33-34, and 37-39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Parady, as applied above.

39. Referring to claim 6, Parady has taught in a processor that is segmented into first portion comprising at least one functional unit (Fig.1, components 38, 40, 42, etc.) and second portion comprising at least one other functional unit (Fig.1, component 32), a method of intelligent caching comprising:

a) in said first portion, executing a first program comprising instructions that manipulate architectural register operands. Note the floating-point operations are executing by the first portion, which includes floating-point functional units. Also, note that they access floating-point registers 50.

b) in said second portion, executing a second program tightly coupled to said first program, said second program comprising an architectural register set switch command and at least one parallel data transfer command that causes data to be transferred between a parallel-loadable register file and a row of a memory array. Note from the abstract that load instructions are the switch commands. That is, loads are actually "load, if cache-miss, switch threads" instructions. Also, load/store unit 32 clearly executes load instructions which load the parallel loadable register file (note that files 48 and 50 in Fig.1 have multiple write ports so that multiple registers may be loaded).

c) wherein said second program monitors at least one bit of information generated during execution of said first program and executes said architectural register set switch command and said parallel data transfer command in support of said first program. Note that the system will

monitor the L2 cache miss signal 114 (Fig.3). If there is a miss, then the switch is actually executed. Also, the parallel data transfer (load) will ultimately be performed. Applicant has not specified when the instruction is performed, so clearly, if a load is to be performed, it will be performed at some point.

d) Parady has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady's main storage to be a DRAM array.

40. Referring to claim 14, Parady has taught an intelligent cache-based processor comprising:

a) a memory array comprising a plurality of random access memory cells. See Fig.2 and note that the system is coupled to main memory which inherently has rows and columns of memory cells. Parady has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady's main storage to be a DRAM array.

b) at least one functional unit. See Fig.1, components 34-46.

c) at least one data assembly unit. See Fig.1, component 32.

d) Parady has taught that said at least one functional unit executes a first program using instructions (functional units execute instructions), but has not taught that said data assembly unit executes an intelligent caching program that causes at least one row of said DRAM array to be speculatively precharged in support of the first program. However, Official Notice is taken that prefetching and its advantages are well known and expected in the art. More specifically, prefetching allows for the retrieval of data from memory before it is needed by an instruction. Consequently, when the instruction finally needs the data, it has already been fetched and is therefore ready immediately. This results in faster execution and higher throughput. And, it is inherent that a DRAM must be precharged before it is read. Similarly, if a prefetch from DRAM is performed, then the DRAM must be speculatively precharged. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady to speculatively precharge and then prefetch from DRAM so that performance may be increased.

41. Referring to claim 15, Parady has taught a processor as described in claim 14. Parady has further taught:

a) first and second dual-port registers files, whereby the first port of each of said register files is a parallel access port and coupled in parallel to said DRAM array, and the second port of each respective register file is coupled said functional unit. See Fig.1, and note that each register file 48 and 50 is a dual port file (has port for reading and port for writing). The first port would be the write port which is coupled to memory for loading data into the file. It would also be a parallel access port because the write port is in fact 3 write ports, according to Fig.1, so 3 reads may occur in parallel. The second port would be the read port so that functional units may read data and operate on it.

b) said at least one functional unit is switchably coupled to said register files and said at least one functional unit executes at least one command to operate on one or more architectural register operands that map onto registers within at least one of said register files. See Fig.3, components 48 and 50, and column 2, lines 25-39, each thread has its own register file. Consequently, when a first thread is active, its register file is also active, while the others are inactive. However, when a second thread becomes active and the first thread becomes inactive due to a switch (see the abstract), the second thread's register file will become active while the rest are inactive.

42. Referring to claim 16, Parady has taught a processor as described in claim 15. Parady has further taught that the data assembly unit is responsive to an instruction set that comprises a command to transfer data in parallel between a row of said DRAM array and a selected one of said register files. The data assembly unit (Fig.1, component 32) is a load/store unit which is responsive to load/store instructions. Load instructions are used to load a register file, and an entire row will be loaded in parallel as each row would comprise multiple bits of data and each bit is written in parallel.

43. Referring to claim 17, Parady has taught a processor as described in claim 16. Parady has further taught that said command to transfer data in parallel between a row of said DRAM array and a selected one of said register files transfers data to or from said speculatively precharged DRAM row. Recall that it would have been obvious to prefetch within Parady because it increases performance. A prefetch is nothing more than a speculative load (the command). Consequently, during a prefetch, data would be transferred from the speculatively charged DRAM row (recall it is inherent that a DRAM must be precharged prior to reading).

44. Referring to claim 18, Parady has taught a processor as described in claim 17. Parady has further taught that said command to transfer data in parallel between a row of said DRAM array and a selected data register file is executed to speculatively prefetch a said DRAM row into a selected one of said register files. Clearly, a prefetch must be executed (performed), and the prefetch will only occur for a selected one of the register files as only one register file is active at any given time.

45. Referring to claim 19, Parady has taught an intelligent cache-based processor comprising:

a) a memory array comprising a plurality of random access memory cells. See Fig.2 and note that the system is coupled to main memory which inherently has rows and columns of memory cells. Parady has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady's main storage to be a DRAM array.

b) at least one functional unit. See Fig.1, components 34-46.

c) at least one data assembly unit. See Fig.1, component 32.

d) first and second dual-port registers files, whereby the first port of each of said register files is a parallel access port and coupled in parallel to said DRAM array, and the second port of each respective register file is coupled said functional unit. See Fig.1, and note that each register file 48 and 50 is a dual port file (has port for reading and port for writing). The first port would be the write port which is coupled to memory for loading data into the file. It would also be a

parallel access port because the write port is in fact 3 write ports, according to Fig.1, so 3 reads may occur in parallel. The second port would be the read port so that functional units may read data and operate on it.

e) Parady has taught that said at least one functional unit executes a first program using instructions (functional units execute instructions), but has not taught that said data assembly unit executes an intelligent caching program that causes at least one row of said DRAM array to be speculatively prefetched into a selected one of said register files in support of the first program. However, Official Notice is taken that prefetching and its advantages are well known and expected in the art. More specifically, prefetching allows for the retrieval of data from memory before it is needed by an instruction. Consequently, when the instruction finally needs the data, it has already been fetched and is therefore ready immediately. This results in faster execution and higher throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady to speculatively precharge and then prefetch from the DRAM so that performance may be increased.

46. Referring to claim 21, Parady has taught an intelligent cache-based processor comprising:

a) a memory array comprising a plurality of random access memory cells. See Fig.2 and note that the system is coupled to main memory which inherently has rows and columns of memory cells. Parady has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been

obvious to one of ordinary skill in the art at the time of the invention to modify Parady's main storage to be a DRAM array.

b) at least one functional unit. See Fig.1, components 34-46.

c) at least one data assembly unit. See Fig.1, component 32.

d) first and second dual-port registers files, whereby the first port of each of said register files is a parallel access port and coupled in parallel to said DRAM array, and the second port of each respective register file is coupled said functional unit. See Fig.1, and note that each register file 48 and 50 is a dual port file (has port for reading and port for writing). The first port would be the write port which is coupled to memory for loading data into the file. It would also be a parallel access port because the write port is in fact 3 write ports, according to Fig.1, so 3 reads may occur in parallel. The second port would be the read port so that functional units may read data and operate on it.

e) whereby said at least one functional unit is configured to execute a first program using instructions involving register operands, and said data assembly unit is configured to execute an intelligent caching program that causes data to be moved between the register files and the DRAM array in support of the first program. Clearly, the system of Fig.1 would operate in such a manner. For instance, component 34 of Fig.1 could execute a first program containing a multiplication operation of two operands (retrieved from the register file). And, the data assembly unit (Fig.1, component 32) will perform loads which move data from memory to the register file in support of a program, i.e., the program will operate on data retrieved from registers which may have been loaded from the DRAM.

47. Referring to claim 22, Parady has taught a processor as described in claim 21. Parady has further taught that the at least one functional unit executes instructions that exclusively include register operands. See Fig. 1, and note for instance that component 34 will execute multiply and divide instructions. These instructions typically include only register operands and are in the form of MULT R3, R2, R1 (multiply R1 and R2 and store the result in R3).

48. Referring to claim 23, Parady has taught a processor as described in claim 21. Parady has further taught that the data assembly unit is responsive to an instruction set that comprises a command to perform the parallel transfer data between a row of said DRAM array and a selected data register file. The data assembly unit (Fig. 1, component 32) is a load/store unit which is responsive to load/store instructions. Load instructions are used to load a register file, and an entire row will be loaded in parallel as each DRAM row would comprise multiple bits of data and each bit is written in parallel.

49. Referring to claim 25, Parady has taught a processor as described in claim 21. Parady has not explicitly taught that the data assembly unit further comprises a row address register and an instruction set which comprises a command to perform arithmetic on said row address register. However, Official Notice is taken that row address registers (i.e., a register which holds a memory address) is well known and expected in the art. More specifically, registers may be used to hold memory addresses so that the system may achieve different types of addressing modes. Instead of specifying long memory addresses within the instruction, a register merely has to be specified and then the address is retrieved from the register. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to have a row address

register in Parady. Furthermore, registers are operated upon. Consequently, commands are available to perform arithmetic in the row address register.

50. Referring to claim 26, Parady has taught a processor as described in claim 25. Parady has further taught that said instruction set further comprises a command to precharge (activate) a row pointed to by said row address register. It is inherent that a row of DRAM must be precharged before it is read. Consequently, a simple load instruction, which reads a row of memory, will precharge the desired row.

51. Referring to claim 28, Parady has taught a processor as described in claim 21. Parady has further taught:

a) at least one coupling between said at least one functional unit and said at least one data assembly unit. See Fig.1, and note that all components are coupled by wires (bus).

b) whereby information passed across said coupling is used to allow said at least one data assembly unit to track the execution of said first program and execute load and store operations in support thereof. Clearly, the load/store unit along with the decoder (which make up part of the data assembly unit) tracks execution of the program and when a load/store instruction is encountered, that operation will be performed to support the program.

52. Referring to claim 33, Parady has taught in an intelligent cache based processor:

a) at least one memory array comprising rows and columns of random access memory cells. See Fig.2 and note that the system is coupled to main memory which inherently has rows and columns of memory cells. Parady has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of

its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady's main storage to be a DRAM array.

b) at least one functional unit that executes a first program (Fig.1, components 32-46), and a data assembly unit that executes an intelligent caching program in support of said first program (Fig.1, component 32 and 14 at the very least make up a data assembly unit), a method of intelligent caching processing comprising:

c) Parady has not taught in said data assembly unit, causing a parallel-loadable register file to be speculatively loaded in parallel from a DRAM row. However, Official Notice is taken that prefetching and its advantages are well known and expected in the art. More specifically, prefetching allows for the retrieval of data from memory before it is needed by an instruction. Consequently, when the instruction finally needs the data, it has already been fetched and is therefore ready immediately. This results in faster execution and higher throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady to prefetch from DRAM so that performance may be increased.

d) in at least one functional unit, generating a conditional output and based on said condition, conditionally mapping said parallel-loadable register file to a set of architectural registers visible to said at least one functional unit. Note that in response to a load instruction being executed, a cache hit/miss signal will be generated. If a miss occurs then a thread switch will occur (see Parady's abstract). If a switch occurs, then a currently inactive register file will become active and be visible to the functional units.

53. Referring to claim 34, Parady has taught a method as described in claim 33. Parady has further taught that said conditionally mapping is initiated under control of said data assembly unit. Note that the data assembly unit comprises load/store unit 32 (Fig.1). The load/store unit executes load instructions which ultimately cause the cache miss. Therefore, the mapping is initiated by the data assembly unit.

54. Referring to claim 37, Parady has taught a method as described in claim 33. Parady has further taught:

a) utilizing said data assembly unit, causing a DRAM row to be speculatively precharged in support of said first program. Recall that it would have been obvious to prefetch in Parady. In order to prefetch, the DRAM row must be precharged or activated. And this is a speculative precharge because the data is being fetched ahead of time without knowing whether the program will use it or not.

b) utilizing said at least one functional unit, executing a command in said first program that causes a register value to be modified. See Fig.1, components 32-46 and note that each of these units is capable of modifying a register (for instance, component 34 may multiply two registers together and store the result in a register).

c) storing at least said modified register value into said speculatively precharged DRAM row. From Fig.1, it can be seen that load/store unit 32 would execute store instructions for storing register data into the DRAM. The result may be stored in the same row from which data was prefetched (speculatively precharged row) as it is a valid memory row just like every other valid memory row.

55. Referring to claim 38, Parady has taught a method as described in claim 33. Parady has further taught:

a) utilizing said data assembly unit, causing a DRAM row to be speculatively precharged prior to the execution of at least one event that is to be executed in said first program. Recall that it would have been obvious to prefetch in Parady. The idea of prefetching is to retrieve data before an execution event requires the data. Consequently, the DRAM row must be precharged or activated before the execution event. And this is a speculative precharge because the data is being fetched ahead of time without knowing whether the program will use it or not.

b) utilizing said at least one functional unit, executing a command to write at least one word to at least one register in a second register set. See Fig. 1, components 32-46 and note that each of these units is capable of modifying a register (for instance, component 34 may multiply two registers together and store the result in a register).

c) utilizing said data assembly unit, causing at least a portion of said second register set to be written into said precharged DRAM row. From Fig. 1, it can be seen that load/store unit 32 would execute store instructions for storing register data into the DRAM. The result may be stored in the same row from which data was prefetched (speculatively precharged row) as it is a valid memory row just like every other valid memory row.

56. Referring to claim 39, Parady has taught a method as described in claim 33. Parady has further taught:

a) utilizing said at least one functional unit, writing an output to a register in an architectural register file. See Fig. 1, components 32-46 and note that each of these units is capable of

modifying a register (for instance, component 34 may multiply two registers together and store the result in a register).

b) utilizing said data assembly unit, reading said output from said register and using said output as a control input in said intelligent caching program. From Fig.1, it can be seen that load/store unit 32 would execute store instructions for storing register data into the DRAM. Therefore, the register data is retrieved and then stored into DRAM. The store data controls which data is stored in DRAM.

57. Claim 11 is rejected under 35 U.S.C. 103(a) as being unpatentable over Inagami in view of Hennessy and further in view of Hayashi, as applied above.

58. Referring to claim 11, Inagami has taught in a processor comprising at least one memory array that comprises rows and columns of random access memory cells (Fig.1, component 1), at least one functional unit which executes a first program (Fig.1, component 5), and a data assembly unit which executes a second program (Fig.1, components 2-0, 2-1, etc.), said second program being tightly coupled with said first program, and whereby said data assembly unit is operative cause a plurality of data elements to be transferred between a memory row and one or more register files (Fig.1, components VR0, VR1,....VR7) that each include a parallel access port, a method of intelligent caching comprising:

a) Inagami has not explicitly taught dispatching in parallel a first sequence of instructions and a second sequence of instructions for parallel execution. However, see Hennessy, pp.278-280, and note that when a system has separate functional units, each one can execute a different instruction per clock cycle (this is a superscalar system). For instance, see figure 4.26 on page

280 and note the existence of an integer unit and a floating-point unit, which are independent of each other. Consequently, they may execute instructions independently in the same cycle, thereby increasing throughput (executing two instructions per cycle instead of one instruction per cycle yields higher throughput). Similarly, Inagami teaches separate load/store pipes 2 and execution pipes 5, just like the integer unit and floating-point unit of Hennessy is separate. The independence of units allows for concurrent execution. Therefore, because Inagami is suited for concurrent execution, in order to increase throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to concurrently dispatch and execute first and second sequences of instructions.

b) executing the first sequence of instructions on said at least one functional unit, said at least one functional unit operative to process data stored in said at least one register file. See column 4, line 63, to column 5, line 1. Note that arithmetic instruction would be executed here.

c) executing the second sequence of instructions on said data assembly unit, said data assembly unit operative cause data to be transferred data between said at least one register file and said array. See Fig.1 and note that data is transferred between registers and memory via the load/store pipes (data assembly unit).

d) Inagami has not taught that said second sequence of instructions instructs said data assembly unit to speculatively prefetch data in parallel from said array in support of said first sequence of instructions. However, Hayashi has taught such a concept. See the title and claim 1 of Hayashi and note that vector data is prefetched and ultimately written to vector registers. Clearly, prefetching is beneficial because by bringing data into the system from memory before instructions requiring that data are executed, once the instruction does in fact need to execute, the

data is available immediately, thereby preventing a time delay. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to include prefetching capability.

e) Inagami has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.

59. Claim 41 is rejected under 35 U.S.C. 103(a) as being unpatentable over Parady, as applied above, in view of Inagami, as applied above, and further in view of Bissett as applied above.

60. Referring to claim 41, Parady in view of Inagami has taught a processor as described in claim 1.

a) Parady has further taught that concurrently with the execution of the first program, said second program executes said commands to unload and load at least some registers. See column 3, lines 11-17 and note that multiple instructions may be dispatched and executed concurrently by not only floating-point and integer units but also by load/store units for loading/storing registers. Parady has not taught that the loading/unloading occurs with respect to said second subset of registers which are inactive. However, Bissett has taught performing data prefetches in the background. And, this is beneficial because background operations do not influence software

execution, thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady in view of Bissett such that Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread). Furthermore, it should be noted that prefetching data also causes unloading of other data. The data being prefetched inherently will replace some other data.

b) Parady has not taught rearranging at least some data in the registers using register-to-register move commands. However, Official Notice is taken the register-register move operations are well known and expected in the art. These operations at the very least allow for the copying of registers. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement register-register move operations. Note that evidence showing that move operations are well known is supplied below.

c) Parady has further taught causing the second subset to become architectural registers visible to said first portion and available for processing by the first program. As described above, when an instruction in a first thread missed the cache (see the abstract), a thread is switched, thereby also causing a new register file to become active. Note that the first program comprises instructions that execute on floating-point units, for instance. Therefore, the new registers will be available to instructions of the first program.

Allowable Subject Matter

61. Claims 2 and 42 are allowed.

Response to Arguments

62. Applicant's arguments filed on May 16, 2005, have been fully considered but they are not persuasive.

63. Firstly, it is noted that applicant traverses the Official Notice used by the examiner. While applicant is correct that Official Notice was used close to 20 times, it should be realized that this is only because of the large number of claims which repeat features (there were at most 5 unique Official Notices taken). For example, the examiner relied on Official Notice to state that DRAM was a well known type of memory. Because this limitation was repeated in the claims twelve times, twelve separate instances of Official Notice corresponding to DRAM appear in the Office Action.

64. In response to applicant's request that the examiner provide support for Official Notice, the examiner would like to bring the following prior art to applicant's attention:

a) With respect to DRAM being well known, more dense (higher capacity), and being less-expensive than other types of memory such as SRAM, see Hennessy and Patterson, "Computer Architecture - A Quantitative Approach, 2nd Edition," 1996, pp. 428-429 (herein referred to as Hennessy). Note that the capacity of DRAM is roughly 4-8 times larger than that of SRAM while SRAM is 8 to 16 times as expensive as SRAM.

b) With respect to move instructions, applicant admits that move instructions are well known in the art on page 23, lines 7-8, of the remarks, so a reference supporting the Official Notice for

move instructions is not provided by the examiner. Applicant's issue with this Official Notice is that move instructions are not well known in the context of applicant's invention, i.e., an embedded DRAM processor that interfaces directly to DRAM. However, the examiner asserts that Parady anticipates the context set forth by applicant because applicant claims "An embedded-DRAM processor..." in the preamble of the claim which does not breathe life into the claim. Therefore, "embedded-DRAM processor" has not been given patentable weight because the recitation occurs in the preamble. A preamble is generally not accorded any patentable weight where it merely recites the purpose of a process or the intended use of a structure, and where the body of the claim does not depend on the preamble for completeness but, instead, the process steps or structural limitations are able to stand alone. See *In re Hirao*, 535 F.2d 67, 190 USPQ 15 (CCPA 1976) and *Kropa v. Robie*, 187 F.2d 150, 152, 88 USPQ 478, 481 (CCPA 1951).

c) With respect to prefetching being well known in the art, see Miura et al., U.S. Patent No. 5,345,560 (hereafter Miura). Note from column 1, lines 12-43, that a prefetch buffer is used to hold prefetched data ahead of when it is needed by the system in order to eliminate overhead due to accessing main memory.

d) With respect to row address registers being well known in the art, see Chia et al., U.S. Patent No. 5,321,425, and note from column 4, lines 33-35, that row address registers are known components for holding addresses used to index memory such as DRAM.

65. Regarding applicant's arguments for claim 1:

a) Applicant argues that the load/store unit (second portion) cannot access any of the deactivated register sets (page 24, lines 12-17). However, it is noted that the second portion accessing a deactivated register set is not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

b) Applicant also argues that there is a cache between the load/store unit (second portion) and any existing memory, and consequently, the load/store unit cannot be called the data assembly unit (page 24, lines 18-23). However, it is not clear what applicant is arguing as there is no "data assembly unit" in claim 1.

c) At the bottom of page 24, applicant requests the examiner to resolve an inconsistency between multiple statements. The examiner asserts that the statement quoted by applicant on page 24, lines 25-28, of the remarks, actually is a part of the rejection of claim 24 (a new paragraph number indicating the rejection of claim 24 was accidentally left out in the initial Office Action). Consequently, when looking at claim 24 and its parent claims, it was established that it would have been obvious for the memory in Parady to be a DRAM. Then, Inagami was used to teach the memory operations on the memory of Parady (hence making them DRAM operations).

d) On page 25, lines 5-9, applicant argues that none of the cited prior art references address the problem of improving performance of a processor that interfaces directly to a DRAM without a need for a caching system. However, claim 1 does not require that there be no caching system.

e) On page 25, lines 10-14, applicant wants evidence showing a register move unit that allows data to be arranged in an inactive register file while other functional units operate on activated architectural registers. The examiner asserts that such a limitation is not in the claim and

therefore the examiner never took Official Notice that such a unit existed. From the rejection of claim 1, the examiner took Official Notice that move operations are well known, and evidence is not provided because applicant admits they are well known (see above).

66. Regarding applicant's arguments for claim 4:

a) Applicant argues on page 27, lines 26-29, that Parady does not teach dispatching in parallel the first sequence of instruction and a second sequence of instructions. However, Parady does in fact teach such a limitation. See Fig.1 and column 3, lines 11-17, and note that multiple instructions are dispatched concurrently to multiple functional units (which include the load/store unit, floating-point units, etc.).

b) Applicant argues on pages 28-29 that Bissett teaches data prefetching in the "context of allowable asynchronous operations in a quasi-synchronous parallel processing system and that instead of teaching or suggesting the Applicant's invention wherein different portions of an architecture work in harmony, one preparing data for (and in advance of the needs of the other portions), Bissett teaches a system where all parallel processors execute the exact same sequence of instructions and operate in a quasi-lock-step. That is, each processor executes the same set of instructions, but not in instruction-level lock step, yet then resynchronize at synchronization points." However, the examiner asserts that applicant's current claim language is not enough to separate itself from Parady. Prefetching itself is a known concept in the art. It allows data to be fetched before it is needed so that when it is in fact needed, the system doesn't need to wait for the data. Bissett is relied on merely to show that prefetching may be done in the background,

which is advantageous because functions in the foreground (current thread execution) are not interrupted.

67. Regarding applicant's arguments for claim 12:

a) Applicant argues on page 29, lines 8-11, that the load/store unit of Parady cannot rightfully be called the data assembly unit because in Parady, a cache exists between the load/store unit and any existing memory. However, it is not clear to the examiner why this would prevent the load/store unit from being a "data assembly unit." More specifically, the load/store unit is still coupled to the DRAM and if the appropriate data to be loaded, for instance, does not exist within the cache, then the data would be retrieved from main memory, as is known in the art of memory hierarchies.

b) Applicant argues on page 29, lines 25-29, that Bissett has been mischaracterized for the same reasons set forth in the arguments for claim 4. Consequently, the examiner's response to this argument is the same as the response to applicant's claim 4 argument with respect to Bissett.

68. Regarding applicant's arguments for claim 29:

a) Applicant argues on page 29, lines 20-24, that the load/store unit of Parady can only interact with active registers, it is coupled to a cache and not a DRAM, and it performs thread switches when an L2 miss occurs, and because of this, Parady cannot compare to applicant's invention. However, the examiner asserts that in addition to be coupled to the cache, it is also coupled to the main memory (DRAM being an obvious variation of memory). In addition, although it only

interacts with active registers in Parady, Bissett was used to modify the unit to perform loads in the background (i.e., to registers not being used by the current thread).

b) Applicant argues on page 29, lines 25-29, that Bissett has been mischaracterized for the same reasons set forth in the arguments for claim 4. Consequently, the examiner's response to this argument is the same as the response to applicant's claim 4 argument with respect to Bissett.

69. Regarding applicant's arguments for claim 7:

a) Applicant argues on page 30, lines 6-10, that Parady does not teach splitting a single program comprises splitting into first and second parallely dispatched and executed portions. However, Parady does in fact teach such a limitation. See Fig.1 and column 3, lines 11-17, and note that multiple instructions are dispatched concurrently to multiple functional units (which include the load/store unit, floating-point units, etc.).

b) Applicant argues on pages 30, lines 11-15, that Bissett does not teach data prefetches in the background but rather a system where al parallel processors execute the exact same sequence of instructions and operate in a quasi-lock-step. However, the examiner asserts that applicant's current claim language is not enough to separate itself from Parady. Prefetching itself is a known concept in the art. It allows data to be fetched before it is needed so that when it is in fact needed, the system doesn't need to wait for the data. Bissett is relied on merely to show that prefetching may be done in the background, which is advantageous because functions in the foreground (current thread execution) are not interrupted. Just because parallel processors may perform differently does not mean that prefetching in the background is not useful in a different system (Parady).

70. Regarding applicant's arguments for claim 6:

a) Applicant argues on page 30, lines 25-31, that the load/store unit of Parady cannot rightfully be called the data assembly unit because in Parady, a cache exists between the load/store unit and any existing memory. However, it is not clear to the examiner why this would prevent the load/store unit from being a "data assembly unit." More specifically, the load/store unit is still coupled to the DRAM and if the appropriate data to be loaded, for instance, does not exist within the cache, then the data would be retrieved from main memory, as is known in the art of memory hierarchies.

b) On page 31, lines 1-4, applicant argues that Parady does not teach a DRAM processor nor a method of intelligent caching for such a processor ("Claim 6 is specifically designed to provide an intelligent caching interface to slower DRAM"). The examiner agrees that Parady has not taught a DRAM, and consequently, Official Notice was taken that DRAM is an advantageous type of memory (Parady has taught memory). In addition, all of applicant's arguments are directed towards language in the preamble. Therefore, the recitation "intelligent caching" has not been given patentable weight because the recitation occurs in the preamble. A preamble is generally not accorded any patentable weight where it merely recites the purpose of a process or the intended use of a structure, and where the body of the claim does not depend on the preamble for completeness but, instead, the process steps or structural limitations are able to stand alone. See *In re Hirao*, 535 F.2d 67, 190 USPQ 15 (CCPA 1976) and *Kropa v. Robie*, 187 F.2d 150, 152, 88 USPQ 478, 481 (CCPA 1951). Furthermore, even if "intelligent caching" was given patentable weight, Parady has taught caching. One of ordinary skill would recognize that

caching in Parady is done with "intelligence" because unintelligent caching would seem to indicate that if reduces efficiency of the processor, which is to be clearly avoided.

c) Finally, applicant argues on page 31, lines 5-9, that the prior art presented does not teach or suggest speculative prefetch of a row of DRAM, and for this reason, claim 6 is novel and non-obvious. However, the examiner asserts that there is no limitation in claim 6 which requires prior art showing a speculative prefetch of a row of DRAM. Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

71. Regarding applicant's arguments for claim 14:

a) Applicant argues on page 31, lines 15-24, in substance that:

"The Examiner basically attempts to say that prefetching is known' and precharging is inherent in DRAM, and therefore it would be obvious to modify Parady to include a speculative prefetch including precharge of DRAM. Aside from the fact that Parady (as well as the other art cited) admittedly does not teach DRAM (see first Official Notice of two used to reject this Claim), it is nonsensical to then assert that such an approach would be combinable with Parady's architecture, let alone suggested or motivated. Note that Parady's load/store unit can only interact with active registers, and is coupled to L2 and L2 cache (not DRAM). This teaches away on multiple bases from a data assembly unit that speculatively fetches one or more rows of DRAM into a register file, along with the other limitations as set forth in Applicant's Claim 14."

In response, the examiner asserts that Parady does not teach DRAM, but that it would have been obvious for the memory of Parady to be a DRAM for advantageous reasons set forth in the rejection. Once Parady is modified to include a DRAM, the examiner asserted that it would have been obvious to prefetch from the DRAM. And, in order to prefetch (read data from DRAM), the DRAM must be precharged. The examiner feels that this is a logical combination. In addition, the load/store unit of Parady is coupled to the DRAM through the L2 cache, but it is

nevertheless coupled. If data to be fetched from memory is not in the cache, it will be retrieved from DRAM. Finally, claim 14 mentions nothing about interacting with inactive register files.

b) Applicant argues on page 31, lines 25-27, that there is no intelligent caching program as set forth by applicant. However, these arguments are directed towards language in the preamble. Therefore, the recitation "intelligent caching" has not been given patentable weight because the recitation occurs in the preamble. A preamble is generally not accorded any patentable weight where it merely recites the purpose of a process or the intended use of a structure, and where the body of the claim does not depend on the preamble for completeness but, instead, the process steps or structural limitations are able to stand alone. See *In re Hirao*, 535 F.2d 67, 190 USPQ 15 (CCPA 1976) and *Kropa v. Robie*, 187 F.2d 150, 152, 88 USPQ 478, 481 (CCPA 1951). Furthermore, even if "intelligent caching" was given patentable weight, Parady has taught caching. One of ordinary skill would recognize that caching in Parady is done with "intelligence" because unintelligent caching would seem to indicate that if reduces efficiency of the processor, which is to be clearly avoided. Applicant has in no way defined how "intelligent caching" separates the claim from the prior art.

72. Regarding applicant's arguments for claim 19:

a) Applicant argues on page 32, lines 8-14, that the load/store unit of Parady cannot rightfully be called the data assembly unit because in Parady, a cache exists between the load/store unit and any existing memory. However, it is not clear to the examiner why this would prevent the load/store unit from being a "data assembly unit." More specifically, the load/store unit is still coupled to the DRAM and if the appropriate data to be loaded, for instance, does not exist within

the cache, then the data would be retrieved from main memory, as is known in the art of memory hierarchies.

b) Regarding the argument that Parady has not taught an embedded DRAM processor, the language is recited in the preamble and is not given patentable weight. However, even if it was, Parady has taught a processor, Parady has taught a memory (DRAM being an obvious variation thereof), and processors are inherently embedded into systems. Furthermore, applicant has not defined what intelligent caching is and how it is different from Parady. If applicant says intelligent caching is performed by speculatively prefetching data from DRAM, and the prior art of record perform such a task, then the prior art performs intelligent caching. It should be realized that the examiner took Official Notice that prefetching is known in the art. In addition, "prefetching" and "speculatively prefetching" are synonymous. The examiner is not clear how these are different. Both phrases imply that data is to be fetched ahead of time even if it isn't used (i.e., the system speculates that data is going to be used in the near future, and so it prefetches it).

73. Regarding applicant's arguments for claim 21, the examiner's responses to these arguments are the same as the responses to the arguments for claim 19 above.

74. Regarding applicant's arguments for claim 33, the examiner's responses to these arguments are the same as the responses to the arguments for claim 19 above. Note that evidence of prefetching being well known was supplied above.

75. Regarding applicant's arguments for claim 40, the examiner agrees that Parady has not taught loading inactive register files. However, prefetching itself is a known concept in the art. It allows data to be fetched before it is needed so that when it is in fact needed, the system doesn't need to wait for the data. Bissett is relied on merely to show that prefetching may be done in the background (i.e., to register files not currently being used), which is advantageous because functions in the foreground (current thread execution) are not interrupted.

b) Furthermore, applicant argues on page 34, lines 23-29, that the load/store unit of Parady cannot rightfully be called the data assembly unit because in Parady, a cache exists between the load/store unit and any existing memory. However, it is not clear to the examiner why this would prevent the load/store unit from being a "data assembly unit." More specifically, the load/store unit is still coupled to the DRAM and if the appropriate data to be loaded, for instance, does not exist within the cache, then the data would be retrieved from main memory, as is known in the art of memory hierarchies.

c) Regarding the argument that Parady has not taught an embedded DRAM processor, the language is recited in the preamble and is not given patentable weight. However, even if it was, Parady has taught a processor, Parady has taught a memory (DRAM being an obvious variation thereof), and processors are inherently embedded into systems. Furthermore, applicant has not defined what intelligent caching is and how it is different from Parady. If applicant says intelligent caching is performed by speculatively prefetching data from DRAM, and the prior art of record perform such a task, then the prior art performs intelligent caching. It should be realized that the examiner took Official Notice that prefetching is known in the art. In addition, "prefetching" and "speculatively prefetching" are synonymous. The examiner is not clear how

these are different. Both phrases imply that data is to be fetched ahead of time even if it isn't used (i.e., the system speculates that data is going to be used in the near future, and so it prefetches it). It is also not clear what evidence applicant wants submitted. The examiner has provided evidence that DRAM is a well-known and advantageous type of memory since this is the only item the examiner took Official Notice on in this claim.

76. Regarding applicant's arguments for claim 11:

a) Applicant argues on page 35, lines 11-20, that Inagami has not taught parallel dispatching a first and second sequence of instructions for execution. The examiner agrees, and consequently, Hennessy has been added to chow that parallel dispatch/execution among independent units is known. So, load/store pipes 2 and execution units 5 in Inagami could take advantage of parallel execution. It should be realized that a first sequence would comprise instructions that are executable by units 5 while a second sequence would comprise instructions that are executable by units 2 (of Inagami).

b) Applicant argues on page 35, lines 23-25, that Hayashi does not perform speculative prefetching of data to avoid delays, but deals with delays that occur after the fact. However, the examiner asserts that the whole idea of prefetching is to fetch data ahead of when it is needed to that it is available immediately when it is needed. This avoids delays.

c) Applicant argues on page 35, lines 26-29, that the prior art has not taught an embedded DRAM processor. However, this language appears in the preamble and is given no patentable weight. Even if it was given weight, Parady has taught a processor, Parady has taught a memory (DRAM being an obvious variation thereof), and processors are inherently embedded into

systems. The combination would be an embedded DRAM processor. The examiner is not clear as to what evidence applicant needs submitted. The examiner has submitted evidence showing that DRAM is well-known and advantageous. No further evidence is needed as the examiner only took Official Notice that DRAM is well known in the art.

Conclusion

77. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
July 19, 2005

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100